F.0 Overview and Expected Functionality

**“The time has come, the Walrus said, to speak of many things,
Of shoes and ships and sealing wax, of cabbages and kings,
Of why the sea is boiling hot and whether pigs have wings”
– L. Carroll**

Richard Weatherly, Ph.D.
21 August 1996

Agenda

- **Discuss current work on the F.0 API**
  - **General form**
  - **Control flow**
  - **Examples**
- **Review expected functionality of F.0**
  - **Which services are in and which are for later**
  - **Test Federate**
  - **Simple Federation Management Tool**
- **Discuss the use of CORBA in F series**

F.0 Application Programmer's Interface Overall Structure

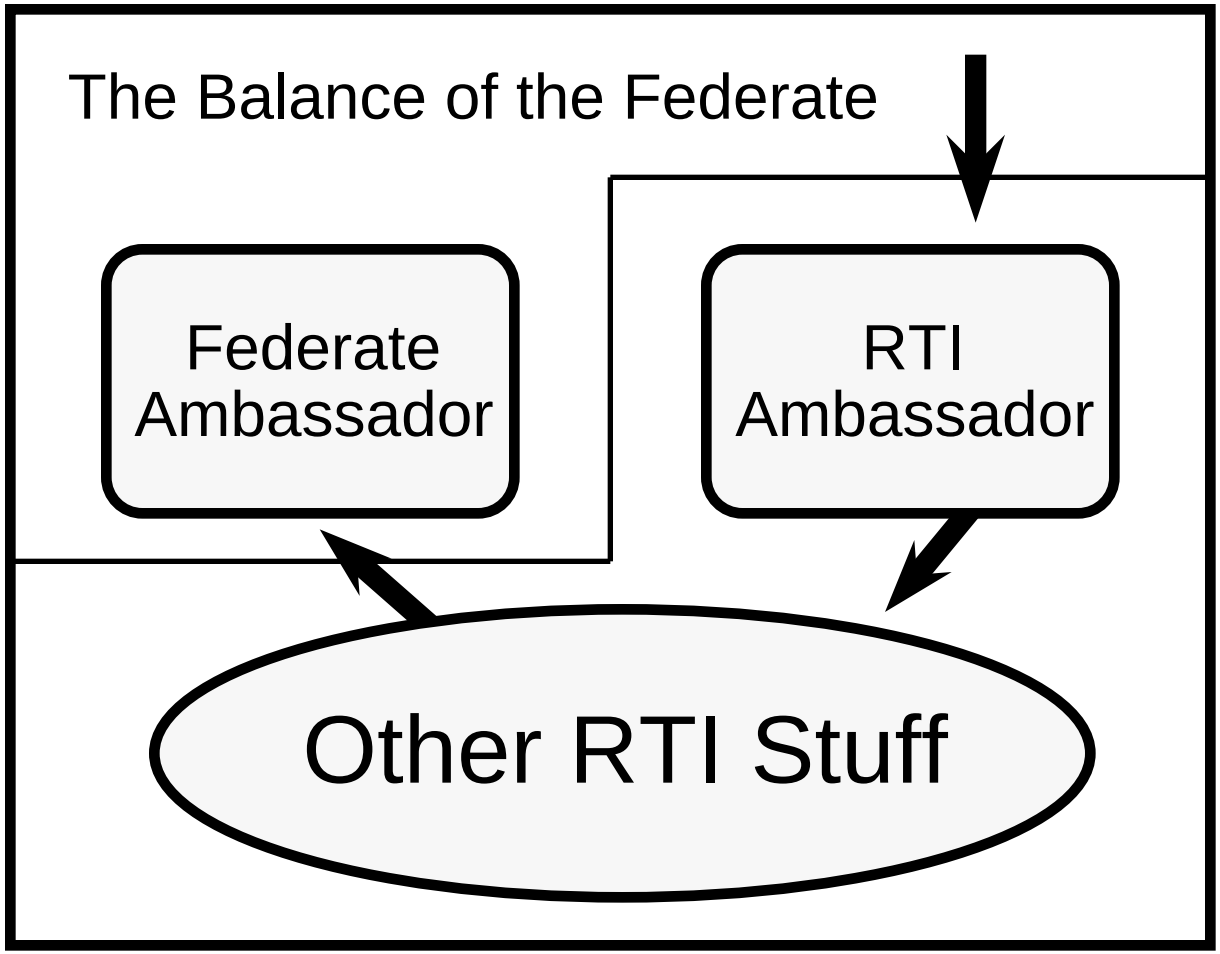
```
class RTI {
public:

  #include "baseTypes.hh
  #include "RTItypes.hh"
  struct RTIambPrivate;

  class RTIambassador {
  public:
    #include "RTIambServices.hh"
  private:
    RTIambPrivate* privateData;
  };

  class FederateAmbassador {
  public:
    #include "federateAmbServices.hh"
  };
};
```

F.0 Flow of Control



Sample Service joinFederationExecution

```
// 2.3
FederateHandle                                          // returned C3
joinFederationExecution (
  const FederateName             yourName,              // supplied C4
  const FederationExecutionName executionName,          // supplied C4
  FederateAmbassadorPtr      federateAmbassadorReference) // supplied C1
throw (
  FederateAlreadyExecutionMember,
  FederationExecutionDoesNotExist,
  ConcurrentAccessAttempted,
  RTIinternalError,
  UnimplementedService);
```

Memory Management Documentation

```
//                RTI F.0 Parameter Passing Memory Conventions
//
// C1  In parameter by value.
// C2  Out parameter by reference.
// C3  Function return by value.
// C4  In parameter by const reference.  Caller provides memory.
//     Caller may free memory or overwrite it upon completion of
//     the call.  Callee must copy during the call anything it
//     wishes to save beyond completion of the call.  Parameter
//     type must define const accessor methods.
// C5  Out parameter by reference.  Caller provides reference to object.
//     Callee constructs an instance on the heap (new) and returns.
//     The caller destroys the instance (delete) at its leisure.
// C6  Function return by reference.  Callee constructs an instance on
//     the heap (new) and returns a reference.  The caller destroys the
//     instance (delete) at its leisure.
```

Expected RTI F.0 Functionality

- **Federation Management**
  - ***Create Federation Execution***
  - ***Destroy Federation Execution***
  - ***Join Federation Execution***
  - ***Resign Federation Execution***
- **Declaration Management**
  - **All services supported**
  - **Full class and attribute based publication and subscription**
  - **Discovery class promotion**
  - ***Control Updates* and *Interactions* do not restrict traffic**
- **Object Management**
  - **All services supported**
  - ***Retract* and *Reflect Retract* design that uses constant storage**

Expected RTI F.0 Functionality

- **Ownership Management**
  - **All services supported**
  - **Design enforces “attribute adopted by good families” requirement**
- **Time Management**
  - **All services supported**
  - **Optimistic services *Request Minimum Next Event Time* and *Flush Queue Request* designed**
  - **More thorough treatment of Time Constrained and Time Regulating switches**
- **Data Distribution Management**
  - **Deferred until after F.0**

Use of CORBA in post-F.0 RTIs

- **Pros**
  - **General distributed OS capability**
    - **Remote object invocation**
    - **Data marshaling**
    - **Name servers**
  - **CORBA/Java/WEB is a big part of the future of simulation**
- **Cons**
  - **Adds complexity**
    - **Additional installation**
    - **Source of “non-RTI” errors**
  - **License fees**